Claims

[c1] 1. A method for operating an out-of-order processor in which a rename process is comprised of the pipeline an instruction stream is processed with, the method comprising the steps of:
for detection of a dependency determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction;
generating a no-dependency-signal associated with said current instruction; and
forwarding said signal for exploiting said signal in order to control the processing of said current instruction in order to bypass a portion of the pipeline.

[c2] 2. The method according to claim 1 in which the step of generating a no-dependency signal comprises the steps of:
comparing a plurality of logic target register addresses and the logic source register address of the current instruction, in case of a match; and
generating a dependency-signal for the respective source register.

[c3] 3. The method according to claim 1 further comprising the step of evaluating "valid"of non-architected target registers stored in a storage associated with speculatively calculated instruction result data into the no-dependency-signal generation.

[c4] 4. The method according to claim 1 further comprising the step of applying the method to a mapping-table-based renaming scheme comprising the step of:
addressing a mapping table entry with a logical source register address of said current instruction thus determining the mapped physical target register address;
reading a committed-status-flag in said entry;
comparing the logic target register address and the logic source register address of the current instruction, and in case of a match; and

generating a dependencyfor the respective source register.

[c5] 5. The method according to claim 2 further comprising the step of applying the method to a mapping-table-based renaming scheme comprising the step of:
addressing a mapping table entry with a logical source register address of said current instruction thus determining the mapped physical target register address;
reading a committed-status-flag in said entry;
comparing the logic target register address and the logic source register address of the current instruction, and in case of a match; and
generating a dependency-signal for the respective source register.

[c6] 6. A processing system having means for executing a readable machine language, said readable machine language comprises:
a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction,
a second computer readable code for generating a no-dependency-signal associated with said current instruction, and
a third computer readable code for forwarding said signal for exploiting said signal in order to control the processing of said current instruction in order to bypass a portion of the pipeline.

[c7] 7. The processing system according to claim 6 in which in case of a content-addressable memory (CAM)-based renaming scheme the first computer readable code for determining the dependency of a current instruction comprises a compare logic in which all instructions to be checked for dependency are involved and a post-connected OR gate.

[c8] 8. The processing system according to claim 7 further comprising a plurality of AND gates the input of which comprises the target register "valid-bits" signal and a respective compare logic output signal.

[c9] 9. The processing system according to claim 6 in which the case of a mapping-table-based renaming scheme each mapping table entry comprises an additional instruction-commited flag, and the first computer readable code for determining the dependency of a current instruction comprises a logic for ANDing a selected in which all instructions to be checked for dependency are involved and a post-connected OR gate.

[c10] 10. A computer system having an out-of-order processing system, said computer system executes a readable machine language, said readable machine language comprises:
a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction,
a second computer readable code for generating a no-dependency-signal associated with said current instruction, and
a third computer readable code for forwarding said signal for exploiting said signal in order to control the processing of said current instruction in order to bypass a portion of the pipeline.